Worksheet 1 Arrays, tuples and records Answers

Task 1

1. Write a program to read 6 numbers into an array **numbers[0]** to **numbers[5]**, ouput them in reverse order and then output the total and average.

```
numbers = [0,0,0,0,0,0]
total = 0
for index = 0 to 5
    numbers[index] = input (“Enter next number”)
    total = numbers[index] + total
next index
for index = 5 to 0 step -1
  output numbers[index]
next index
print (total)
average = total/6
print (average)
```

2. A teacher uses a program that stores pupil names in an array. The array is indexed from 0, so the first element in the array is **name[0]**. Occasionally the teacher needs to search for a name to find the student’s record number, which is **index + 1**.

   Write a pseudocode algorithm that will search an array **name** containing **max** elements, to find a name and output record number if it exists. If the name does not exist the user should be told the term was not found. Use appropriate prompts for input and output in your solution.

```
max = 5
name = [“Annie”,”Bob”,”Charles”,”Dan”,”Erica”,Faisal”]
searchName = input ( “Enter search name”)
found = False
for index =  0 to max
    if name[index] == searchName then
        print (“Record number: “, index + 1)
        found = True
    endif
next index
if found == False then
    print (“Search name not found”)
endif
```

3. Sales quantities of a certain item, calculated to the nearest thousand, for Jan-March, April-June, July-Sep and Oct-Dec are held in separate arrays for each of 3 outlets. The sales figures for each quarter are to be totalled and output in the format

   Total for quarter 1 xxxx

   Total for quarter 2 xxxx

   Total for quarter 3 xxxx

   Total for quarter 4 xxxx

   Write a pseudocode algorithm for this program. Initialise the array with the following test data:

   Outlet1 = [10, 12,15,10]

   Outlet2 = [5, 8, 3, 6]

   Outlet3 = [10,12,15,10]

```
outlet1Sales = [10, 12,15,10]
outlet2Sales = [5, 8, 3, 6]
outlet3Sales = [10,12,15,10]
totalSales = [0,0,0,0]
for quarter = 0 to 3
    totalSales[quarter] = totalSales[quarter] + outlet1Sales[quarter] +outlet2Sales[quarter] + outlet3Sales[quarter]
    print ("Total for quarter ", quarter+1, totalSales[quarter])
next quarter
```

Task 2

4. (a) Now suppose, in question 3, there were 50 outlets. Assuming the array **outletSales[4,50]** holds the sales values for each quarter for each outlet, complete the following algorithm to output the total sales figures for each quarter.

```
Fill array outletSales with sales values
Initialise each element of array total[4] to zero

for quarter = 0 to 3

    for outlet = 0 to 49

        total[quarter] = total[quarter]  + outletSales[quarter,outlet]
    next outlet

    print ("Total for quarter ",quarter+1, total[quarter])
    next quarter
```

5. A grid game draws a 6 by 4 grid with each square denoted by “x”. A character “O” can move by entering a row coordinate from 1 to 6 and column co-ordinate from 1 to 4. The character starts at array position [0,0] (Figure 1) and will move, for example, to row 0 column 1 (Figure 2) if the user enters 1, 2 for the row and column coordinates. **Remember that the indices of the array both start at 0**.

   Write a pseudocode algorithm that creates a 2-D grid[row,column], drawn as shown in Figure 1.

   Prompt the user to enter a row and column value. Update the character position and draw the new grid.

```
O x x x
x x x x
x x x x
x x x x
x x x x
x x x x
```

Figure 1

```
x O x x
x x x x
x x x x
x x x x
x x x x
x x x x
```

Figure 2

```
for row = 0 to 5
    for column = 0 to 3
        grid[row, column] = “x”
    next column
next row
grid[0, 0] = ”O”
row = input (“Enter column to move to”)
row = row – 1                                // adjust for rows 1 to 6, not 0 to 5
column = input (“Enter column to move to”)
column = column – 1                          // adjust for columns 1 to 4, not 0 to 3

grid[0,0] = “x”
grid[row,column] = ”O”
for row = 0 to 5
    for column = 0 to 3
        print (grid[row,column] , “  ”)          // print without moving to a new
line
    next column
    print ( “”)                                  // this is to provide a newline
next row
```

6. A company runs a private car park near an airport. The car park has 10 rows numbered 1-10 and each row has spaces (referred to as columns) numbered 1-6 for 6 cars. Customers leave their cars with keys at the car park office, and a driver parks it in a free space and then records where it is parked.
The space is referenced by its grid coordinates row and column. E.g. a car parked in the 3rd row, 5th space would have the grid reference [3, 5].

   The driver enters the car registration into the computer. A car with registration AVH 61 HU parked at grid reference [3, 5] would assign "AVH 61 HU" to **park[3, 5]**. Empty spaces are denoted, for example, by **park[3, 5] = "empty"**

   Write pseudocode for a program which :

   Initialises the grid, with each element holding "empty".

   "Parks a car". This option asks the user to enter the registration number of a car and the grid reference (row and column number) where it has been parked.

   Validates the user entry row between 1 and 10, column between 1 and 6 and asks user to re-enter until entry is valid.

   Checks that this is an empty space, and if it is, puts the registration number in the appropriate element of the array. If it is not, displays "That space is taken" and asks the user to re-enter the grid reference.

   Displays the grid.

   Note that the way the 2-D array is initialised will vary in different programming languages – the pseudocode does not go into details. Indices start at 0. Program solutions in Python and VB are provided.

```
Initialise the car park grid to "empty"
(For test purposes, set carpark[0, 0] to "TAKEN")
Display grid
prompt for and enter car registration
prompt for and enter row and column where car is parked
emptySpace = False
while emptySpace == False
    while row < 1 or row > 10
        row = input ("Row must be between 1 and 10 - please re-enter: ")
    endwhile
    row = row - 1
    rowValid = True
    while column < 1 or column > 6
        column = input ("Column must be between 1 and 6 - please re-enter: ")
    endwhile
    column = column - 1
    columnValid = True
    if rowValid and columnValid and carPark[row, column] == "empty" then
        emptySpace = True
    else
        rowValid = False
        columnValid = False
```

```
            print ("That space is taken")
        endif
    endwhile
    carPark[row, column] = regNo

    #print the grid
    for row = 0 to 9
        for column = 0 to 5
            symbol = carPark[row, column]
            print (symbol, " ", end = "")                    // print without moving to
    new line
        next column
        print ()                                        // move to new line
    next row
```

Task 3

7. An application is plotting points in 3D space, using x, y, and z coordinates. Explain how the coordinates could be stored in each of the following and justify if the structure is a good choice for this task.

   (a) 1D array

   A 1D array could be used where all the x, y, and z coordinates can be stored one after another. To access each point, a step of 3 will be needed, and you would need to know how many points are to be plotted. This would work, but is not the best choice.

   (b) 2D array

   In a 2-D array each row of 3 elements could represent one point and each column the x, y and z coordinates respectively. P(0,0) will hold the x coordinate for point 0, P(0,1) the y coordinate and P(0,3) the z coordinate.This would be a good choice if you know how many points are to be plotted, or at least the maximum number.

   (c) 3D array

   Three dimensions are not needed for the array. You will never need to refer to a point with coordinates (x_1, y_6, z_3), for example. This is a poor choice.

   (d) tuple

   As in the 1D array, to access each point a step of 3 will be needed and you would need to know how many points are to be plotted, since tuples are immutable. This would work, but is not the best choice.

   (e) list of tuples

   Each x, y, and z combination could be a tuple, represented as (x, y, z). The list would be ((x,y,z), (x,y,z), …). This is a good choice.

   (f) 1D array of tuples

   An array has to be of fixed length, so you need to know how many points are to be plotted and therefore how many tuples you have. Then, each item in the array is of the same type, namely a tuple. This is a good choice.

   (g) array of records

   Records would work because each record would have 3 fields, each of the same type, which is fine. An array would be a fixed size. This is almost conceptually the same as a list of tuples. This is a good choice.

(h) file of records

Each line of the file could be three numbers, separated by a space or a comma, ending with a line feed. Each line could be read in and the x, y, and z pulled from the line. This is a good choice.